*Project AVES*



Table of Content

1.      Introduction

2.      Requirements

3.      Functionality Description

Introduction

Will finish abstract intro paragraph later

           Project *AVES* is a program that calculates optimization of a single flight pattern at a specific instance in time.

Unlike - future implementation (real time market ex: fuel cost, airline competitors, season)

Primary differentiation - future implementation

**Vision:** For any airline that needs to optimize seating per aircraft, the AViation Estimation Software (AVES) is a software application that finds the most profitable way to fill a flight.

Disclaimers:

* AVES is not prepared to perform dynamic optimizations
  + example: fuel cost, stock market, season
* AVES performs optimizations for an instance in time (no real-time calculations)
  + Assumes that the input list of ticket requested for *a* airplanes that can fly *f* times at *p* airports is finalized - meaning real-time cancellation or purchases will not be considered for now
* AVES assumes that all ticket prices are fixed
  + No difference in price for One-Way and Multiple Stop tickets
* AVES will calculate timezones using Coordinated Universal Time (UTC)
* AVES will only optimize domestic flights

fix distance, schedule and prices

have multiple stop discount incentive over  (when a person cannot have a nonstop flight)

price of tickets are set by the market (competitors are smaller factors to consider later), number of people. If there are too many people that want a particular flight, increase the price. If there is a surplus of empty seats, lower to cost as an incentive to fill up seats

Constraints need to be placed such that prices will not increase infinitely or infinite multi-stops are made such that they are virtually costless OR same price as a Non-stop flight

Profit is determined by # of seats sold - fuel cost

Requirements

Precondition for Database:

1. Every Person is associated with an Account
2. Tickets purchased will be formatted in UTC

(Draw Logic Flow later)

Purchase Ticket via Airline Website ->

Database ->

List of Request Tickets (output from Database) ->

List<Ticket> -> AVES Application -> Final List<Ticket>

Displays on Graphical Interface for Users at Airline

English Definition of each Object in AVES:

Each ***Account***:

* contains a billing information
  + Address
  + Zip\*
  + Email
  + Gender
  + Phone\*
  + Name
  + DoB
* encapsulates 1 to many ***Persons***

Each ***Person***:

* contains personal information
  + Name
  + DoB
  + Gender
* is requests/ owns 0 to many tickets

Each ***Ticket (abstract base class)***:

* is associated with 1 ***Person***
* is associated with 1 ***Flight***

(ONE-WAY and MULTISTOP)?

(CLASSES of Tickets)

Each ***Plane (abstract base class)***:

* has a FAA\_Num (Plane ID number)
* has a model numbers (public static string)
* has seat capacity of X
  + can have 1 to many classes of seatings (list of)
* is associated with 0 to many ***Flights***

Each ***Airport***:

* has a *unique* FAA\_LID (Airport Identifier for domestic flights)
* services X amounts of ***Airplanes*** (limited number of ports allocated for each Airline)

Each ***Flight***:

* is associated with an ***Airplane*** (FAA\_NUM)
* is associated with exactly two ***Airports*** (FAA\_LID: start\_destination and end\_destination)
* is associated with exactly two different times (UTC: Start, End)

September 18

Accomplished:

* goal/ vision
* abstract model/ system diagram
* defined core objects

September 19

Paul

* interface IAircraft (Airplane, Helicopter, missile, spaceship are all *types* of Aircraft)
* controller main()
* abstract class Airplane

September 21

Aleks

* Accounts
* Persons